Titre
Modules sémantiques robustes pour une réutilisation saine en DL-lite

Auteurs
François Goasdoué
Univ. Paris-Sud & CNRS (LRI/IASI) – INRIA (Saclay/Leo)
INRIA, Parc Orsay Université, 4 rue J. Monod, 91893 Orsay Cedex, France
fg@lri.fr

Marie-Christine Rousset
Univ. Grenoble & CNRS (LIG/HADAS) – INRIA (Saclay/Webdam)
LIG, 681 rue de la passerelle, BP 72, 38402 St Martin d'Hères Cedex, France
Marie-Christine.Rousset@imag.fr

Résumé
De plus en plus de systèmes de gestion de données modernes sont conçus à l'aide d'ontologies. Les ontologies permettent de structurer les données d'un domaine d'application – provenant éventuellement de multiples sources – afin de faciliter leur interrogation.

Dans de nombreux domaines d'application, des ontologies standardisées (quasi-)exhaustives résultant d'initiatives collaboratives sont mises à disposition. De telles ontologies sont souvent associées avec des données fiables – collectées et vérifiées avec soin –, fournissant ainsi des systèmes de gestion de données de référence dans ces domaines d'application.

Pour construire un nouveau système adapté à des besoins particuliers, une bonne pratique est alors de partir d'un système de référence préexistant, d'en extraire le fragment (appelé module) pertinent vis-à-vis des besoins, et d'enrichir ce fragment avec les données et connaissances propres à l'application développée.

L'objet de cet article est la définition et l'étude de la *réutilisation saine* d'un module afin de construire un système de gestion de données consistant vis-à-vis d'un système de gestion de données de référence – pour bénéficier de la qualité de son schéma et de ses données –, de façon à ce qu'il puisse être interrogé seul ou conjointement au système de référence – pour bénéficier de ses données et augmenter le nombre de réponses aux requêtes –.

Mots clés
Logiques de description, Modules, Web Sémantique

Robust Semantic Modules for Safe Reuse in DL-lite

François Goasdoué* Marie-Christine Rousset†

Abstract

More and more modern data management systems are defined using ontologies. Ontologies provide the basis for structuring data of a given application domain, possibly coming from multiple sources, in order to facilitate their interrogation. In many application domains, comprehensive ontologies resulting from collaborative initiatives are made available. Such well-established ontologies are often associated with reliable data – carefully collected and verified –, thus providing reference data management systems in different domains of application. To build a new system with specific needs, it is therefore a good practice to start from an existing reference system, to extract a fragment (called a module) that is relevant to our needs, and then to enrich this fragment with new data and knowledge proper to our own exploitation needs.

The purpose of this paper is to define and study the *safe reuse* of modules for building data management systems in a way guaranteeing their consistency with respect to a reference system, so that they can be queried either independently or jointly with the reference system.

1 Introduction

More and more modern data management systems are defined using ontologies. An ontology is a formal description providing human users a shared understanding of a given domain. The ontologies we consider here can also be interpreted and processed by machines thanks to a logical semantics that enables reasoning. Ontologies provide the basis for sharing knowledge and as such, they are very useful for structuring data of a given application domain, possibly coming from different sources, and for facilitating their uniform interrogation.

*LRI: Univ. Paris-Sud & CNRS – INRIA Saclay – fg@lri.fr

†LIG: Univ. of Grenoble & CNRS – INRIA Saclay– Marie-Christine.Rousset@imag.fr

In many application domains (e.g., medecine or biology), comprehensive ontologies resulting from collaborative initiatives are made available. For instance, SNOMED is a medical ontology containing more than 400.000 concept names covering various areas such as anatomy, diseases, medication, and even geographic locations. Such well-established ontologies are often associated with reliable data that have been carefully collected and verified, thus providing reference data management systems in different domains of application.

For building a new data management system with specific needs, it is therefore a good practice to start from an existing reference data management system, to extract a fragment (called a module) that is relevant to our needs, and then to enrich this fragment with new data and knowledge proper to our own exploitation needs. By doing so, not only we can reuse the part of a reference system that is of interest for us, but we can combine the reference system with the newone in order to obtain enriched answers or to validate our local answers against the reference system. This requires that the new data management system is developed and evolves coherently w.r.t. the reference data management system.

The purpose of this paper is to define and study the *safe reuse* of modules for building data management systems in a way guaranteing their consistency with respect to a reference data management system, so that they can be queried either independently or jointly with the reference data management system.

The problem of extracting modules from ontologies has been recently studied for different Description Logics (DLs) which underlie modern ontology languages, like OWL2[1] from W3C. Slightly different definitions have been proposed for a module w.r.t. a subset of the vocabulary in the initial ontology, which corresponds to the atomic concepts and roles of interest for the application under construction. Those definitions are based on the tightly connected notions of deductive conservative extension [7, 15] and of uniform interpolation [10, 11].

A module can be a *subset* of the initial ontology or the result of *forgetting* from the initial ontology the atomic concepts or roles that are not useful in a particular application setting. The problem of forgetting is a reasoning problem which has been studied in classical logic [14] and recently addressed in DLs. For instance, both kinds of modules have been investigated in DL-lite [12, 19], $\mathcal{EL}$ [9, 10, 11], and $\mathcal{ALC}$ [5, 9, 18].

In this paper, we define a novel notion of *semantic modules*, which captures both the modules obtained by extracting a subset of a Tbox or by

[1] http://www.w3.org/TR/owl2-overview/

forgetting concepts or roles.

We then define and study the *safe reuse* of a semantic module of a given global Tbox for building local Aboxes, and querying them either independently or jointly with the global Abox. For enabling the local Abox (associated to the module) and the global Abox (associated to the original Tbox) to evolve independently while coherently, we generalize the notion of query conservative extension defined in [13] and we extend it to consistency checking.

Finally, we provide algorithms and complexity results for the computation of minimal and robust semantic modules in DL-lite$_{\mathcal{F}}$ and DL-lite$_{\mathcal{R}}$. Those dialects are members of the DL-Lite family [3] which has been specially designed for querying efficiently large datasets. Notably, DL-lite$_{\mathcal{R}}$ provides the foundations of the QL profile of OWL2[2] which extends RDFS[3] (the W3C recommendation for writing simple ontologies) with interesting constructors such as inverse roles and disjointness between concepts and between roles.

2 Preliminaries

DLs [2] underly modern ontology languages such as the W3C recommendations for the Semantic Web. An ontology formulated in a DL $\mathcal{L}$ is a set of terminological axioms in $\mathcal{L}$ called a *Tbox*. This represents the background knowledge about a factual dataset called an *Abox*.

A Tbox $\mathcal{T}$ is defined upon a *signature* (or *vocabulary*), denoted $\mathsf{sig}(\mathcal{T})$, which is the disjoint union of a set of *atomic concepts* representing sets of elements, and a set of *atomic roles* representing binary relations between elements.

An Abox defined upon $\mathsf{sig}(\mathcal{T})$ is a set of *assertional* axioms on *constants* representing elements of the domain of interest.

A knowledge base (KB) $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is made of a *Tbox* $\mathcal{T}$ representing a conceptual view of the domain of interest (i.e., an ontology) and an *Abox* $\mathcal{A}$ representing the data (i.e., a local set of facts).

The legal KBs vary according to the DL $\mathcal{L}$ used for defining complex concepts and roles, and to the restrictions that are stated on the allowed terminological or assertional axioms. In Section 4, we will focus on KBs for DL-lite.

DL-lite KBs. In DL-lite, the concepts and roles that can be built from atomic concepts and roles are of the following form:

[2]http://www.w3.org/TR/owl2-profiles/

[3]http://www.w3.org/TR/rdf-schema/

$B \rightarrow A \mid \exists R,\ C \rightarrow B \mid \neg B,\ R \rightarrow P \mid P^-,\ E \rightarrow R \mid \neg R$
where A denotes an *atomic concept*, P an *atomic role*, and P^- the *inverse* of P. B denotes a *basic concept* (i.e., an atomic concept A or an *unqualified existential quantification on a basic role* $\exists R$) and R a *basic role* (i.e., an atomic role P or its inverse P^-). Finally, C denotes a *general concept* (i.e., a basic concept or its negation) and E a *general role* (i.e., a basic role or its negation).

The semantics of concepts and roles is given in terms of interpretations. An *interpretation* $I = (\Delta^I, .^I)$ consists of a nonempty *interpretation domain* Δ^I and an *interpretation function* $.^I$ that assigns a subset of Δ^I to each atomic concept, and a binary relation over Δ^I to each atomic role. The semantics of non atomic concepts and roles is defined as follows:

- $(P^-)^I = \{(o_2, o_1) \mid (o_1, o_2) \in P^I\}$,
- $(\exists R)^I = \{o_1 \mid \exists o_2\ (o_1, o_2) \in R^I\}$, and
- $(\neg B)^I = \Delta^I \backslash B^I$ and $(\neg R)^I = \Delta^I \times \Delta^I \backslash R^I$.

The axioms allowed in a Tbox of DL-lite are concept inclusion statements of the form $B \sqsubseteq C$. DL-lite$_\mathcal{F}$ and DL-lite$_\mathcal{R}$ are two dialects of DL-lite that differ from some additional allowed axioms: a DL-lite$_\mathcal{F}$ Tbox allows functionality statements on roles of the form (*funct* R), while a DL-lite$_\mathcal{R}$ Tbox allows role inclusion statements of the form $R \sqsubseteq E$. It is worth noticing that general concepts or roles are only allowed on the right hand side of inclusion statements whereas only basic concepts or roles may occur on the left hand side of such statements. Moreover, functionality statements are only allowed on basic roles. Inclusions of the form $B_1 \sqsubseteq B_2$ or $R_1 \sqsubseteq R_2$ are called *positive inclusions (PIs)*, while inclusions of the form $B_1 \sqsubseteq \neg B_2$ or of the form $R_1 \sqsubseteq \neg R_2$ are called *negative inclusions (NIs)*. A PI allows expressing that a basic concept (or role) subsumes another one, while a NI allows expressing that two basic concepts (or roles) are disjoint.

An interpretation $I = (\Delta^I, .^I)$ is a *model of an inclusion* $B \sqsubseteq C$ (resp. $R \sqsubseteq E$) if $B^I \subseteq C^I$ (resp. $R^I \subseteq E^I$). It is a *model of a functionnality statement* (*funct* R) if the binary relation R^I is a function, i.e., $(o, o_1) \in R^I$ and $(o, o_2) \in R^I$ implies $o_1 = o_2$. I is a *model of a Tbox* if it is a model of all of its statements. A Tbox is *satisfiable* if it has a model. A Tbox $\mathcal{T}$ *logically entails* a statement α, written $\mathcal{T} \models \alpha$, if every model of $\mathcal{T}$ is a model of α.

An Abox consists of a finite set of membership assertions of the form $A(a)$ and $P(a, b)$, i.e., on atomic concepts and roles, stating respectively that a is an instance of A and that the pair of constants (a, b) is an instance of P. The interpretation function of an interpretation $I = (\Delta^I, .^I)$ is extended to

constants by assigning to each constant a a distinct object $a^I \in \Delta^I$ (i.e., the so called *unique name assumption* holds). An interpretation I is a *model of the membership assertion* $A(a)$ (resp. $P(a,b)$) if $a^I \in A^I$ (resp., $(a^I, b^I) \in P^I$). It is a *model of an Abox* if it satisfies all of its assertions.

An interpretation I is a *model of a KB* $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ if it is a model of both $\mathcal{T}$ and $\mathcal{A}$. A KB $\mathcal{K}$ is *satisfiable* (a.k.a. *consistent*) if it has at least one model. A KB $\mathcal{K}$ *logically entails* a statement or assertion β, written $\mathcal{K} \models \beta$, if every model of $\mathcal{K}$ is a model of β.

Queries over a KB. A query q is of the form $q(\bar{x}) : \phi(\bar{x}, \bar{y})$ where $\phi(\bar{x}, \bar{y})$ is a FOL (first-order logic) formula, the variables of which are *only* the free variables $\bar{x}$ and the bound variables $\bar{y}$, and the predicates of which are either *atomic* concepts or roles of the KB. The *arity* of a query is the number of its free variables, e.g., 0 for a *boolean query*. When $\phi(\bar{x}, \bar{y})$ is of the form $\exists \bar{y}\ conj(\bar{x}, \bar{y})$ where $conj(\bar{x}, \bar{y})$ is a conjunction of atoms, q is called a *conjunctive query*. Conjunctive queries, a.k.a. select-project-join queries, are the core database queries.

Given an interpretation $I = (\Delta^I, .^I)$, the semantics q^I of a boolean query q is defined as true if $[\phi(\emptyset, \bar{y})]^I = \text{true}$, and false otherwise, while the semantics q^I of a query q of arity $n \geq 1$ is the relation of arity n defined on Δ^I as follows: $q^I = \{\bar{e} \in (\Delta^I)^n \mid [\phi(\bar{e}, \bar{y})]^I = \text{true}\}$. An interpretation that evaluates a boolean query to true, respectively a non boolean query to a non empty set, is a model of that query.

Answers of a query over a KB. Let q be a query and a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$.

If q is non boolean, the answer set of q over $\mathcal{K}$ is defined as: $ans(q, \mathcal{K}) = \{\bar{t} \in \mathcal{C}^n \mid \mathcal{K} \models q(\bar{t})\}$ where $\mathcal{C}$ is the set of the constants appearing in the KB, $q(\bar{t})$ is the closed formula obtained by replacing in the query definition the free variables in $\bar{x}$ by the constants in $\bar{t}$, and $\mathcal{K} \models q(\bar{t})$ obviously means that every model of $\mathcal{K}$ is a model of $q(\bar{t})$.

If q is boolean, the answer set of q over $\mathcal{K}$ is by convention either $\{\langle\rangle\}$, where $\langle\rangle$ is the empty tuple, or $\emptyset$ otherwise. More specifically, $ans(q, \mathcal{K}) = \{\langle\rangle\}$ if and only if $\mathcal{K} \models q()$, i.e., every model of $\mathcal{K}$ is a model of $q()$.

Answering by query reformulation. The DL-Lite family [3] has been designed for answering *conjunctive* queries in two steps:

1. the query reformulation algorithm $\mathsf{PerfectRef}(q, \mathcal{T})$ computes the most general conjunctive queries which, together with the axioms in the Tbox $\mathcal{T}$, entail the query;

2. each of those query reformulations is then evaluated against the Abox seen as a relational database.

Separating the data processing from the ontology reasoning is very important both from a practical and a theoretical perspective. Such an approach has the practical interest that it makes possible to use an SQL engine for the second step, thus taking advantage of well-established query optimization strategies supported by standard relational data management systems. From a theoretical point of view, this gives a bound on the data complexity of query answering, since it is known that evaluating FOL queries over a relational database is in LogSpace (more precisely in AC_0) in the size of the database [17, 1]. In particular, it results from the P-completeness (shown in [3]) of instance checking in DL-lite knowledge bases expressed in the union of DL-lite$_\mathcal{F}$ and DL-lite$_\mathcal{R}$ that the completeness of the answers obtained by query reformulation is not guaranteed in slight extensions of DL-lite$_\mathcal{F}$ or DL-lite$_\mathcal{R}$. This is why we focus on DL-lite$_\mathcal{F}$ and DL-lite$_\mathcal{R}$ in the algorithmic part of this paper (Section 4). In fact, those algorithms apply to Tboxes expressed in the union of DL-lite$_\mathcal{F}$ and DL-lite$_\mathcal{R}$ in which the functionality statements do not concern roles involved in role inclusion statements. Such a restriction on the union of DL-lite$_\mathcal{F}$ and DL-lite$_\mathcal{R}$, which provides the foundations of DL-lite$_\mathcal{A}$ [16], enables to keep the FOL reducibility of queries, and thus the completeness of query answering by reformulation.

3 Problem statement

Our definition of *semantic module* is closely related to that of deductive conservative extension (e.g., [7]). For a given signature, it captures *all* the knowledge of an original Tbox. Notably, it generalizes the modules defined either as a subset of the original Tbox or as the result of forgetting all the atomic concepts and roles of the original Tbox that are not of interest.

Definition 1 (Semantic module) *Let $\mathcal{T}$ be a Tbox and $\Gamma \subseteq \mathsf{sig}(\mathcal{T})$. A* semantic module *of $\mathcal{T}$ w.r.t. Γ is a Tbox $\mathcal{T}_\Gamma$ such that:*

- $\mathcal{T} \models \mathcal{T}_\Gamma$ *and*
- *for any Tbox statement α defined upon Γ, $\mathcal{T} \models \alpha$ iff $\mathcal{T}_\Gamma \models \alpha$.*

It is worth noticing that the above definition *implies* that $\Gamma \subseteq \mathsf{sig}(\mathcal{T}_\Gamma) \subseteq \mathsf{sig}(\mathcal{T})$, provided that $\mathcal{T}$ and $\mathcal{T}_\Gamma$ are satisfiable. We will denote by $\mathsf{sig}^+(\mathcal{T}_\Gamma)$ the set difference $\mathsf{sig}(\mathcal{T}_\Gamma)\backslash\Gamma$ between the signature of $\mathcal{T}_\Gamma$ and Γ.

Example (Running example) Consider the signature $\Gamma = \{\texttt{Plant}, \texttt{HasDNA}\}$ and the DL-lite KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ such that $\mathcal{T} = \{\texttt{Plant} \sqsubseteq \texttt{LivingOrganism}$, $\texttt{Human} \sqsubseteq \texttt{LivingOrganism}, \texttt{Human} \sqsubseteq \neg\texttt{Plant}, \texttt{LivingOrganism} \sqsubseteq \exists\texttt{HasDNA}$, $\exists\texttt{HasDNA} \sqsubseteq \texttt{LivingOrganism}\}$ and $\mathcal{A} = \{\texttt{Human}(a)\}$.

Both $\mathcal{T}^1_\Gamma = \{\texttt{Plant} \sqsubseteq \exists\texttt{HasDNA}\}$ and $\mathcal{T}^2_\Gamma = \{\texttt{Plant} \sqsubseteq \texttt{LivingOrganism}$, $\texttt{LivingOrganism} \sqsubseteq \exists\texttt{HasDNA}\}$ are semantic modules of $\mathcal{T}$ w.r.t. Γ, with $\mathcal{T}^1_\Gamma \not\subseteq \mathcal{T}$ and $\mathcal{T}^2_\Gamma \subseteq \mathcal{T}$. We have $\mathsf{sig}^+(\mathcal{T}^1_\Gamma) = \emptyset$, while $\mathsf{sig}^+(\mathcal{T}^2_\Gamma) = \{\texttt{LivingOrganism}\}$. Note that $\mathcal{T}^1_\Gamma$ is the result of forgetting $\mathsf{sig}(\mathcal{T})\backslash\Gamma$ in $\mathcal{T}$, while it is not the case for $\mathcal{T}^2_\Gamma$. ■

The purpose of extracting a module of a Tbox w.r.t. a signature is to reuse it in a local setting for managing data (i.e., an Abox) w.r.t. that signature. When the original Tbox is associated with an Abox, forming a reference KB $\mathcal{K}$, we want to help users to develop *independently* their local KB while keeping it *consistent* with the original KB. For instance in the above example, if the local Abox $\{\texttt{Plant}(a)\}$ is associated with the semantic module $\mathcal{T}^1_\Gamma$ (or $\mathcal{T}^2_\Gamma$), leading to a consistent local KB $\mathcal{K}'$, we would like to detect the global inconsistency with the original KB, *without having to build* $\mathcal{K} \cup \mathcal{K}'$.

In addition, we also want to make possible to *complete* the answers of a query posed locally on $\mathcal{K}'$ by exploiting the knowledge of the original KB $\mathcal{K}$. For instance, while there is no answer to the query $q(x) : \exists y\ \texttt{HasDNA}(x, y)$ posed locally to $\mathcal{K}' = \langle \mathcal{T}^1_\Gamma, \emptyset \rangle$ (or $\mathcal{K}' = \langle \mathcal{T}^2_\Gamma, \emptyset \rangle$), it may be useful to get the answer a from $\mathcal{K} \cup \mathcal{K}'$, again *without having to build* $\mathcal{K} \cup \mathcal{K}'$.

For that purpose, we define two notions of *robustness* for semantic modules. From now on, $\mathcal{A}_{/\mathsf{sig}}$ denotes the restriction of an Abox $\mathcal{A}$ to the assertions of $\mathcal{A}$ built upon the signature sig only.

A semantic module defined w.r.t. a signature Γ is *robust to consistency checking* (Definition 2), if it contains enough knowledge to infer inconsistency related to the signature Γ from any Abox associated with the *initial* Tbox.

Definition 2 (Robustness to consistency checking) *Let $\mathcal{T}_\Gamma$ be a semantic module of a Tbox $\mathcal{T}$ w.r.t. a signature $\Gamma \subseteq \mathsf{sig}(\mathcal{T})$. $\mathcal{T}_\Gamma$ is* robust to consistency checking *iff for every Abox $\mathcal{A}_\Gamma$ made of statements involving Γ and for every Abox $\mathcal{A}_{\bar{\Gamma}}$ built upon $\mathsf{sig}(\mathcal{T})\backslash\Gamma$ and consistent with $\mathcal{T}$: $\langle \mathcal{T}, \mathcal{A}_\Gamma \cup \mathcal{A}_{\bar{\Gamma}} \rangle$ is consistent iff $\langle \mathcal{T}_\Gamma, (\mathcal{A}_\Gamma \cup \mathcal{A}_{\bar{\Gamma}})_{/\mathsf{sig}(\mathcal{T}_\Gamma)} \rangle$ is consistent.*

Proposition 1 shows how global consistency can be checked by accessing locally to $\mathcal{A}_{/\mathsf{sig}(\mathcal{T}_\Gamma)}$.

Proposition 1 (Global consistency checking) *If $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is a consistent KB and $\mathcal{K}' = \langle \mathcal{T}_\Gamma, \mathcal{A}' \rangle$ is a KB such that $\mathcal{T}_\Gamma$ is a semantic module of $\mathcal{T}$ w.r.t. Γ, $\mathcal{T}_\Gamma$ is robust to consistency checking, and $\mathcal{A}'$ is built upon Γ, then: $\langle \mathcal{T}, \mathcal{A} \cup \mathcal{A}' \rangle$ is consistent iff $\langle \mathcal{T}_\Gamma, \mathcal{A}_{/\mathsf{sig}(\mathcal{T}_\Gamma)} \cup \mathcal{A}' \rangle$ is consistent.*

Proposition 1 follows from the fact that $\langle \mathcal{T}, \mathcal{A} \cup \mathcal{A}' \rangle \models \langle \mathcal{T}_\Gamma, (\mathcal{A} \cup \mathcal{A}')_{/\mathsf{sig}(\mathcal{T}_\Gamma)} \rangle$ and $(\mathcal{A} \cup \mathcal{A}')_{/\mathsf{sig}(\mathcal{T}_\Gamma)} = \mathcal{A}_{/\mathsf{sig}(\mathcal{T}_\Gamma)} \cup \mathcal{A}'$, for one direction. For the converse direction, consider the subset $(\mathcal{A} \cup \mathcal{A}')_{\bar{\Gamma}}$ of $\mathcal{A} \cup \mathcal{A}'$ built upon $\mathsf{sig}(\mathcal{T}) \backslash \Gamma$. We thus have $(\mathcal{A} \cup \mathcal{A}')_{\bar{\Gamma}} \subseteq \mathcal{A}$. Since $\mathcal{K}$ is consistent and $\mathcal{K} \models \langle \mathcal{T}, (\mathcal{A} \cup \mathcal{A}')_{\bar{\Gamma}} \rangle$, $\langle \mathcal{T}, (\mathcal{A} \cup \mathcal{A}')_{\bar{\Gamma}} \rangle$ is consistent. As a result, Definition 2 applies and since $\langle \mathcal{T}_\Gamma, (\mathcal{A} \cup \mathcal{A}')_{/\mathsf{sig}(\mathcal{T}_\Gamma)} \rangle$ is consistent and $(\mathcal{A} \cup \mathcal{A}')_{/\mathsf{sig}(\mathcal{T}_\Gamma)} = \mathcal{A}_{/\mathsf{sig}(\mathcal{T}_\Gamma)} \cup \mathcal{A}'$, we get: $\langle \mathcal{T}, \mathcal{A} \cup \mathcal{A}' \rangle$ is consistent.

A semantic module defined w.r.t. a signature Γ is *robust to query answering* (Definition 3) if it contains enough knowledge to infer all the answers of any query built upon Γ asked to any (global) Abox associated with the *initial* Tbox. This generalizes the notion of query conservative extension defined in [13] which considers queries asked to any Abox built upon Γ *only*. As for the forgetting technique introduced in [19, 18], it allows a property similar to that of robustness to query answering *under the (severe) condition* that the initial Abox is modified.

Definition 3 (Robustness to query answering) *Let $\mathcal{T}_\Gamma$ be a semantic module of a Tbox $\mathcal{T}$ w.r.t. a signature $\Gamma \subseteq \mathsf{sig}(\mathcal{T})$. $\mathcal{T}_\Gamma$ is* robust to query answering *iff for every Abox $\mathcal{A}$ consistent with $\mathcal{T}$ and for every query q built upon Γ: $ans(q, \langle \mathcal{T}, \mathcal{A} \rangle) = ans(q, \langle \mathcal{T}_\Gamma, \mathcal{A}_{/\mathsf{sig}(\mathcal{T}_\Gamma)} \rangle)$.*

Proposition 2 shows how global query answering can be done by accessing locally to $\mathcal{A}_{/\mathsf{sig}(\mathcal{T}_\Gamma)}$.

Proposition 2 (Global query answering) *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a consistent KB and $\mathcal{K}' = \langle \mathcal{T}_\Gamma, \mathcal{A}' \rangle$ a KB such that $\mathcal{T}_\Gamma$ is a semantic module of $\mathcal{T}$ w.r.t. Γ which is robust to both consistency checking and query answering, and $\mathcal{A}'$ is built upon Γ. Let q be a query built upon Γ. If $\langle \mathcal{T}_\Gamma, (\mathcal{A} \cup \mathcal{A}')_{/\mathsf{sig}(\mathcal{T}_\Gamma)} \rangle$ is consistent then: $ans(q, \langle \mathcal{T}, \mathcal{A} \cup \mathcal{A}' \rangle) = ans(q, \langle \mathcal{T}_\Gamma, \mathcal{A}_{/\mathsf{sig}(\mathcal{T}_\Gamma)} \cup \mathcal{A}' \rangle)$.*

Proposition 2 first follows from Proposition 1 to get the consistency of $\langle \mathcal{T}, \mathcal{A} \cup \mathcal{A}' \rangle$. As a result, Definition 3 applies and since $(\mathcal{A} \cup \mathcal{A}')_{/\mathsf{sig}(\mathcal{T}_\Gamma)} = \mathcal{A}_{/\mathsf{sig}(\mathcal{T}_\Gamma)} \cup \mathcal{A}'$, we get: $ans(q, \langle \mathcal{T}_\Gamma, \mathcal{A}_{/\mathsf{sig}(\mathcal{T}_\Gamma)} \cup \mathcal{A}' \rangle) = ans(q, \langle \mathcal{T}, \mathcal{A} \cup \mathcal{A}' \rangle)$.

Property 1 follows from the fact that a Tbox is a semantic module of itself (robust to both consistency checking and query answering).

Property 1 *A semantic module of a Tbox, possibly robust to query answering and/or consistency checking, always exists.*

For a given Tbox and a given signature, several semantic modules may exist. We define the notion of *minimality* in order to compare them.

Definition 4 (Minimal semantic module) *Let $\mathcal{T}_\Gamma$ be a semantic module of a Tbox $\mathcal{T}$ w.r.t. a signature $\Gamma \subseteq \mathsf{sig}(\mathcal{T})$, which is possibly robust to query answering and/or consistency checking. $\mathcal{T}_\Gamma$ is* minimal *iff for any other semantic module $\mathcal{T}'_\Gamma$ of $\mathcal{T}$ w.r.t. Γ with the same robustness: if $\mathcal{T}_\Gamma \models \mathcal{T}'_\Gamma$ then $\mathcal{T}'_\Gamma \equiv \mathcal{T}_\Gamma$.*

Propositions 1 and 2 show how a semantic module can be reused for (global) data management. We now generalize the *reuse* of a module within a local KB by considering Tbox and Abox updates.

We define the *reuse* of a semantic module $\mathcal{T}_\Gamma$ as a KB made of a Tbox $\mathcal{T}'$ obtained from $\mathcal{T}_\Gamma$ by some updates, and an associated Abox built upon $\mathsf{sig}(\mathcal{T}')\backslash\mathsf{sig}^+(\mathcal{T}_\Gamma)$, i.e., upon what remains of Γ after the updates plus the new atomic concepts and roles introduced by those updates.

Definition 5 (Reuse of a semantic module) *Let $\mathcal{T}$ be a Tbox and let $\mathcal{T}_\Gamma$ be a semantic module of $\mathcal{T}$ w.r.t. $\Gamma \subseteq \mathsf{sig}(\mathcal{T})$. A KB $\mathcal{K}' = \langle \mathcal{T}', \mathcal{A}' \rangle$ is a* reuse *of $\mathcal{T}_\Gamma$ iff $\mathcal{T}'$ is obtained from $\mathcal{T}_\Gamma$ after a sequence of insertions or deletions of Tbox statements and $\mathcal{A}'$ is an Abox built upon $\mathsf{sig}(\mathcal{T}')\backslash\mathsf{sig}^+(\mathcal{T}_\Gamma)$.*

Definition 6 exhibits sufficient conditions for a reuse of a semantic module to be *safe*, i.e., to preserve the possibility to perform locally the global consistency checking and/or the query answering of the reused module together with the reference KB. First, the Tbox updates cannot involve atomic concepts and roles of the reference KB other than those in the reused module; second, the Tbox updates must keep intact the terminological knowledge of the reference KB; third, the resulting updated Tbox must be a semantic module of the global knowledge w.r.t. its data management signature and must have the same robustness(es) as the reused module.

Definition 6 (Safe reuse of a semantic module w.r.t. a KB) *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a KB and let $\mathcal{T}_\Gamma$ be a semantic module of $\mathcal{T}$ w.r.t. $\Gamma \subseteq \mathsf{sig}(\mathcal{T})$. A reuse $\mathcal{K}' = \langle \mathcal{T}', \mathcal{A}' \rangle$ of $\mathcal{T}_\Gamma$ is* safe *w.r.t. $\mathcal{K}$ iff*

1. $\mathsf{sig}(\mathcal{T}) \cap (\mathsf{sig}(\mathcal{T}')\backslash\mathsf{sig}(\mathcal{T}_\Gamma)) = \emptyset$,
2. *$\mathcal{T}$ is a semantic module of $\mathcal{T} \cup \mathcal{T}'$ w.r.t. $\mathsf{sig}(\mathcal{T})$, and*
3. *$\mathcal{T}'$ is a semantic module of $\mathcal{T} \cup \mathcal{T}'$ w.r.t. $\mathsf{sig}(\mathcal{T}')\backslash\mathsf{sig}^+(\mathcal{T}_\Gamma)$ with the same robustness(es) as $\mathcal{T}_\Gamma$.*

Propositions 3 and 4 result directly from Definition 6 by adapting accordingly the proofs of Propositions 1 and 2.

Proposition 3 (Safe global consistency checking) *If $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is a consistent KB and $\mathcal{K}' = \langle \mathcal{T}', \mathcal{A}' \rangle$ a safe reuse of a semantic module $\mathcal{T}_\Gamma$ of $\mathcal{T}$ w.r.t. Γ such that $\mathcal{T}_\Gamma$ is robust to consistency checking, then: $\langle \mathcal{T} \cup \mathcal{T}', \mathcal{A} \cup \mathcal{A}' \rangle$ is consistent iff $\langle \mathcal{T}', \mathcal{A}_{/\mathsf{sig}(\mathcal{T}')} \cup \mathcal{A}' \rangle$ is consistent.*

Proposition 4 (Safe global query answering) *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a consistent KB and $\mathcal{K}' = \langle \mathcal{T}', \mathcal{A}' \rangle$ a safe reuse of a semantic module $\mathcal{T}_\Gamma$ of $\mathcal{T}$ w.r.t. Γ such that $\mathcal{T}_\Gamma$ is robust to both consistency checking and query answering. Let q be a query built upon $\mathsf{sig}(\mathcal{T}') \backslash \mathsf{sig}^+(\mathcal{T}_\Gamma)$. If $\langle \mathcal{T}', \mathcal{A}_{/\mathsf{sig}(\mathcal{T}')} \cup \mathcal{A}' \rangle$ is consistent, then: $ans(q, \langle \mathcal{T} \cup \mathcal{T}', \mathcal{A} \cup \mathcal{A}' \rangle) = ans(q, \langle \mathcal{T}', \mathcal{A}_{/\mathsf{sig}(\mathcal{T}')} \cup \mathcal{A}' \rangle)$.*

The following example points out that safe global consistency checking and safe global query answering depend on the inherited robustnesses of the reused module (*3.* in Definition 6).

Example (continued) The semantic module $\mathcal{T}_\Gamma^1 = \{\texttt{Plant} \sqsubseteq \exists \texttt{HasDNA}\}$ of $\mathcal{T}$ is *not* robust to query answering, since for $q(x) : \exists y\ \texttt{HasDNA}(x, y)$, $a \in ans(q, \langle \mathcal{T}, \mathcal{A} \rangle)$ while $a \notin ans(q, \langle \mathcal{T}_\Gamma^1, \mathcal{A}_{/\mathsf{sig}\mathcal{T}_\Gamma^1} \rangle)$.

Let us consider the global KB $\mathcal{K}_1$ made of $\mathcal{T}$ associated with the Abox $\{\texttt{LivingOrganism}(o), \texttt{Plant}(p)\}$, and the reuse $\mathcal{K}_1' = \langle \mathcal{T}', \mathcal{A}' \rangle$ of $\mathcal{T}_\Gamma^1$ such that $\mathcal{T}' = \{\exists \texttt{HasDNA} \sqsubseteq \neg \texttt{Patentable}\}$ and $\mathcal{A}' = \{\texttt{Patentable}(a), \texttt{HasDNA}(b, c)\}$. While $\mathcal{K}_1'$ is a safe reuse of $\mathcal{T}_\Gamma^1$ w.r.t. $\mathcal{K}_1$, the global answer o to the query $q(x) : \exists y\ \texttt{HasDNA}(x, y)$ cannot be obtained from $\langle \{\exists \texttt{HasDNA} \sqsubseteq \neg \texttt{Patentable}\}, \{\texttt{Patentable}(a), \texttt{HasDNA}(b, c)\} \rangle$, although it is entailed by $\mathcal{K}_1 \cup \mathcal{K}_1'$. ■

Finally, Corollary 1 is a useful corollary of Proposition 4 for DL-lite in which query answering relies on the query reformulation performed by $\mathsf{PerfectRef}(q, \mathcal{T})$, which is a reasoning step independent of the data, as reminded in Section 2. It basically says that the reformulation step for obtaining global answers can be entirely local to $\mathcal{T}'$.

Corollary 1 *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a consistent DL-lite$_\mathcal{F}$ or DL-lite$_\mathcal{R}$ KB and $\mathcal{K}' = \langle \mathcal{T}', \mathcal{A}' \rangle$ is a safe reuse of a semantic module $\mathcal{T}_\Gamma$ of $\mathcal{T}$ w.r.t. Γ such that $\mathcal{T}_\Gamma$ is robust to query answering. Let q be a query built upon $\mathsf{sig}(\mathcal{T}') \backslash \mathsf{sig}^+(\mathcal{T}_\Gamma)$. If $\langle \mathcal{T}', \mathcal{A}_{/\mathsf{sig}(\mathcal{T}')} \cup \mathcal{A}' \rangle$ is consistent, then:*
$ans(q, \langle \mathcal{T} \cup \mathcal{T}', \mathcal{A} \cup \mathcal{A}' \rangle) = ans(\mathsf{PerfectRef}(q, \mathcal{T}'), \langle \emptyset, \mathcal{A}_{/\mathsf{sig}(\mathcal{T}')} \cup \mathcal{A}' \rangle)$.

4 Extracting semantic modules in DL-lite

Our algorithm for extracting modules from a given DL-lite Tbox (Algorithm 1) relies on the notion of (deductive) closure of a Tbox (Definition 7). The

closure of a Tbox basically amounts to exhibit Tbox statements that are logically entailed by that Tbox.

Definition 7 (Closure of a Tbox) *Let $\mathcal{T}$ be a DL-lite$_\mathcal{F}$ or a DL-lite$_\mathcal{R}$ Tbox. We call the* closure *of $\mathcal{T}$, denoted by $cl(\mathcal{T})$, the Tbox defined inductively as follows:*

1. *All the statements in $\mathcal{T}$ are also in $cl(\mathcal{T})$.*
2. *If $B_1 \sqsubseteq B_2$ and $B_2 \sqsubseteq B_3$ are in $cl(\mathcal{T})$, then $B_1 \sqsubseteq B_3$ is in $cl(\mathcal{T})$.*
3. *If $R_1 \sqsubseteq R_2$ and $\exists R_2 \sqsubseteq B$ are in $cl(\mathcal{T})$, then $\exists R_1 \sqsubseteq B$ is in $cl(\mathcal{T})$.*
4. *If $R_1 \sqsubseteq R_2$ and $R_2 \sqsubseteq R_3$ are in $cl(\mathcal{T})$, then $R_1 \sqsubseteq R_3$ is in $cl(\mathcal{T})$.*
5. *If $R_1 \sqsubseteq R_2$ is in $cl(\mathcal{T})$, then $R_1^- \sqsubseteq R_2^-$ is in $cl(\mathcal{T})$.*
6. *If $B_1 \sqsubseteq B_2$ and $B_2 \sqsubseteq \neg B_3$ (or $B_3 \sqsubseteq \neg B_2$) are in $cl(\mathcal{T})$, then $B_1 \sqsubseteq \neg B_3$ is in $cl(\mathcal{T})$.*
7. *If $R_1 \sqsubseteq R_2$ and $\exists R_2 \sqsubseteq \neg B$ (or $B \sqsubseteq \neg \exists R_2$) are in $cl(\mathcal{T})$, then $\exists R_1 \sqsubseteq \neg B$ is in $cl(\mathcal{T})$.*
8. *If $R_1 \sqsubseteq R_2$ and $\exists R_2^- \sqsubseteq \neg B$ (or $B \sqsubseteq \neg \exists R_2^-$) are in $cl(\mathcal{T})$, then $\exists R_1^- \sqsubseteq \neg B$ is in $cl(\mathcal{T})$.*
9. *If $R_1 \sqsubseteq R_2$ and $R_2 \sqsubseteq \neg R_3$ (or $R_3 \sqsubseteq \neg R_2$) are in $cl(\mathcal{T})$, then $R_1 \sqsubseteq \neg R_3$ is in $cl(\mathcal{T})$.*
10. *If $R_1 \sqsubseteq \neg R_2$ or $R_2 \sqsubseteq \neg R_1$ is in $cl(\mathcal{T})$, then $R_1^- \sqsubseteq \neg R_2^-$ is in $cl(\mathcal{T})$.*
11. (a) *In the case in which $\mathcal{T}$ is a DL-lite$_\mathcal{F}$ Tbox, if one of the statements $\exists R \sqsubseteq \neg \exists R$ or $\exists R^- \sqsubseteq \neg \exists R^-$ is in $cl(\mathcal{T})$, then both such statements are in $cl(\mathcal{T})$.*
    (b) *In the case in which $\mathcal{T}$ is a DL-lite$_\mathcal{R}$ Tbox, if one of the statements $\exists R \sqsubseteq \neg \exists R$, $\exists R^- \sqsubseteq \neg \exists R^-$, or $R \sqsubseteq \neg R$ is in $cl(\mathcal{T})$, then all three such statements are in $cl(\mathcal{T})$.*

Proposition 5 exhibits notable properties of the closure of a Tbox in terms of upper bounds of its size and of its computation time. It also characterizes the knowledge made explicit from the Tbox: the closure of a Tbox includes all the strongest statements logically entailed by that Tbox (a.k.a. the prime implicates of the Tbox).

Proposition 5 (Properties of the closure of a Tbox) *Let $\mathcal{T}$ be a DL-lite$_\mathcal{F}$ or a DL-lite$_\mathcal{R}$ Tbox.*

1. *The number of statements in* $cl(\mathcal{T})$ *of is at most quadratic in the size of* $\mathsf{sig}(\mathcal{T})$.

2. $cl(\mathcal{T})$ *can be computed in polynomial time in the size of* $\mathsf{sig}(\mathcal{T})$.

3. $\mathcal{T} \equiv cl(\mathcal{T})$ *and:*

   - *Let* $X \sqsubseteq Y$ *be a NI or a PI. If* $\mathcal{T} \models X \sqsubseteq \neg X$ *then* $X \sqsubseteq \neg X \in cl(\mathcal{T})$*, otherwise* $\mathcal{T} \models X \sqsubseteq Y$ *iff* $X \sqsubseteq Y \in cl(\mathcal{T})$ *or* $\neg Y \sqsubseteq \neg X \in cl(\mathcal{T})$.
   - $\mathcal{T} \models (funct\ R)$ *iff* $(funct\ R) \in cl(\mathcal{T})$ *or* $\exists R \sqsubseteq \neg\exists R \in cl(\mathcal{T})$.

The first item of Proposition 5 directly follows from the statements that are allowed in a DL-lite$_{\mathcal{F}}$ or a DL-lite$_{\mathcal{R}}$ Tbox.

Then, for proving the second item, we consider the items of Definition 7 (except the first one) as *closure rules* that are exhaustively applied to the Tbox, as soon as their conditions can be matched with (original or derived) Tbox statements. The result follows from that (i) each closure rule has atmost 3 conditions, (ii) at each iteration, each condition of each closure rule can be matched to a number of statements which is less than the total number of statements in $cl(\mathcal{T})$ (at most quadratic in the size of $\mathsf{sig}(\mathcal{T})$, as shown above), and (iii) the number of iterations is less than the number of derived statements in $cl(\mathcal{T})$ (at most quadratic in the size of $\mathsf{sig}(\mathcal{T})$).

Finally, for proving the third item, observe first that $\mathcal{T} \equiv cl(\mathcal{T})$ since the closure of a Tbox $\mathcal{T}$ contains all the statements of $\mathcal{T}$, plus statements that are logically entailed by $\mathcal{T}$ (those added by the closure rules). Therefore, if $X \sqsubseteq Y \in cl(\mathcal{T})$ or $\neg Y \sqsubseteq \neg X \in cl(\mathcal{T})$ (respectively $X \sqsubseteq \neg X \in cl(\mathcal{T})$), then $\mathcal{T} \models X \sqsubseteq Y$ (respectively $\mathcal{T} \models X \sqsubseteq \neg X$). Similarly, if $(funct\ R) \in cl(\mathcal{T})$, then $\mathcal{T} \models (funct\ R)$. For the converse way, we consider the (clausal form of) the FOL translation of Tbox statements. We observe that if $\mathcal{T} \models X \sqsubseteq \neg X$, the FOL translation of $\neg X$ is a prime implicate of the FOL theory (that we denote FOL($\mathcal{T}$)) made of the FOL translation of the statements in $\mathcal{T}$. Otherwise, $\mathcal{T} \models X \sqsubseteq Y$ (with $\mathcal{T} \not\models X \sqsubseteq \neg X$) and the FOL translation of $\neg X \vee Y$ is also a prime implicate of FOL($\mathcal{T}$). Then, by exploiting the completeness of resolution for the derivation of primes implicates of FOL theories ([8, 6]), we show by induction on the minimal length l of a derivation of such a prime implicate, that there is a finite sequence of applications of closure rules that adds the DL-lite translation of that prime implicate to $cl(\mathcal{T})$. Lastly, Theorem 25 in [3] states that $\mathcal{T} \models (funct\ R)$ iff $(funct\ R) \in \mathcal{T}$ or $\mathcal{T} \models \exists R \sqsubseteq \neg\exists R$. If $(funct\ R) \in \mathcal{T}$, then $(funct\ R) \in cl(\mathcal{T})$ because of *1.* in Definition 7. Otherwise, $\exists R \sqsubseteq \neg\exists R \in cl(\mathcal{T})$ as shown above.

With the definition of closure in place, we provide the Extract Semantic Module (ESM) algorithm (Algorithm 1) that builds a semantic module of a Tbox w.r.t. a signature and the desired robustness. The resulting module is minimal and is obtained by filtering the closure of the Tbox or the Tbox itself for keeping the relevant statements.

Algorithm 1: the ESM algorithm
$\mathsf{ESM}(\mathcal{T}, \Gamma, RQA, RCC)$
Input: a DL-lite$_\mathcal{F}$ or DL-lite$_\mathcal{R}$ Tbox $\mathcal{T}$, $\Gamma \subseteq \mathsf{sig}(\mathcal{T})$, two booleans RQA and RCC
Output: a semantic module of $\mathcal{T}$ w.r.t. Γ, which is minimal, robust to query answering if $RQA =$ true, and robust to consistency checking if $RCC =$ true
(1) $\mathcal{T}_\Gamma \leftarrow \emptyset$
(2) **foreach** $\alpha \in cl(\mathcal{T})$
(3) **if** α is built upon Γ only
(4) $\mathcal{T}_\Gamma \leftarrow \mathcal{T}_\Gamma \cup \{\alpha\}$
(5) **else if** $RCC =$ true and α is a NI $X \sqsubseteq \neg Y$ s.t. X or Y is built upon Γ
(6) $\mathcal{T}_\Gamma \leftarrow \mathcal{T}_\Gamma \cup \{\alpha\}$
(7) **if** $RQA =$ true
(8) $\mathsf{sig} \leftarrow \Gamma$; $\mathcal{T}'_\Gamma \leftarrow \emptyset$
(9) **while** $\mathcal{T}_\Gamma \neq \mathcal{T}'_\Gamma$
(10) $\mathcal{T}'_\Gamma \leftarrow \mathcal{T}_\Gamma$; $\mathsf{sig}' \leftarrow \mathsf{sig}$
(11) **foreach** PI $X \sqsubseteq Y \in \mathcal{T}$ s.t. Y is built upon sig'
(12) $\mathcal{T}_\Gamma \leftarrow \mathcal{T}_\Gamma \cup \{X \sqsubseteq Y\}$
(13) $\mathsf{sig} \leftarrow \mathsf{sig} \cup \mathsf{sig}_X$ (for X built upon the signature sig_X)
(14)**return** $\mathcal{T}_\Gamma$

Theorem 1 (Properties of the ESM algorithm) *Let $\mathcal{T}$ be a DL-lite$_\mathcal{F}$ or DL-lite$_\mathcal{R}$ Tbox and Γ a subset of $\mathsf{sig}(\mathcal{T})$. $\mathsf{ESM}(\mathcal{T}, \Gamma, RQA, RCC)$ terminates and returns a semantic module of $\mathcal{T}$ w.r.t. Γ, which is minimal, robust to query answering if $RQA =$* true*, and robust to consistency checking if $RCC =$* true*.*

The termination of $\mathsf{ESM}(\mathcal{T}, \Gamma, RQA, RCC)$ follows from the finiteness of $cl(\mathcal{T})$, for lines 2–6, and from that $\mathcal{T} \subseteq \mathcal{T}_\Gamma$ in the worst case, so $\mathcal{T}_\Gamma = \mathcal{T}'_\Gamma$ eventually, for lines 7–13.

$\mathcal{T}_\Gamma$, the output of $\mathsf{ESM}(\mathcal{T}, \Gamma, RQA, RCC)$, is a semantic module of Tbox $\mathcal{T}$ w.r.t. Γ because of $\mathcal{T}_\Gamma \subseteq cl(\mathcal{T})$ and the filtering condition of line 3.

Whenever $RCC =$ true, the robustness to consistency checking is guaranteed by the lines 5–6. Lemma 16 in [3] states that a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is consistent iff every NI or functionality entailed by $\mathcal{T}$ has no counterexample

in $\mathcal{A}$. We therefore get the robustness to consistency checking, since $\mathcal{T}_\Gamma$ contains *all* (and *only*) the NIs and functionalities of $cl(\mathcal{T})$ involving (at least) a relation of Γ.

If RQA = true, the robustness to query answering is guaranteed by the lines 7–13. Theorem 40 (for DL-lite$_\mathcal{R}$) and Theorem 45 (for DL-lite$_\mathcal{F}$) in [3] state that $ans(q, \mathcal{K})$ for a conjunctive query q over a consistent KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ can be achieved by first reformulating q using the PIs in $\mathcal{T}$ into a union of conjunctive queries over $\mathcal{K}$ and then by evaluation the reformulation on $\mathcal{A}$ seen as a relational database. Notably, the reformulation step is performed by the $\mathsf{PerfectRef}(q, \mathcal{K})$ algorithm. From Lemma 39 in [3], which characterizes the output of $\mathsf{PerfectRef}(q, \mathcal{K})$ w.r.t. query answering, we show that the lines 7–13 add to $\mathcal{T}_\Gamma$ *all* (and *only*) the PIs required by $\mathsf{PerfectRef}(q, \mathcal{K})$ w.r.t. query answering, for q built upon Γ.

Finally, when RCC = false and RQA = false, $\mathcal{T}_\Gamma$ is minimal since it is constructed from lines 1–4, thus built upon Γ only. When RCC = true or RQA = true, minimality results from that *only* the required statements are added by lines 5–6 and lines 7–13.

Corollary 2 states the complexity of computing semantic modules.

Corollary 2 (Complexity of computing a semantic module) *Let $\mathcal{T}$ be a DL-lite$_\mathcal{F}$ or DL-lite$_\mathcal{R}$ Tbox and $\Gamma \subseteq \mathsf{sig}(\mathcal{T})$. Computing a semantic module of $\mathcal{T}$ w.r.t. Γ, possibly robust to query answering and/or consistency checking, is polynomial in the size of $\mathsf{sig}(\mathcal{T})$.*

The worst cost computation is that of a semantic module robust to both consistency checking and query answering. $\mathsf{ESM}(\mathcal{T}, \Gamma, \text{true}, \text{true})$ first computes a module robust to consistency checking using lines 1–6. The closure of $\mathcal{T}$ must be computed and then its statements are filtered. This can be done in polynomial time in the size of $\mathsf{sig}(\mathcal{T})$ according to *1.* and *2.* in Proposition 5. Then, $\mathsf{ESM}(\mathcal{T}, \Gamma, \text{true}, \text{true})$ filters $\mathcal{T}$ for robustness to query answering. In the worst case, we have $\mathsf{sig}(\mathcal{T})$ iterations of the **while** loop (since $\mathsf{sig} \subseteq \mathsf{sig}(\mathcal{T})$), each of which has to filter $\mathcal{T}$ whose size is at most quadratic in the size of $\mathsf{sig}(\mathcal{T})$ (since $\mathcal{T} \subseteq cl(\mathcal{T})$). As a result, the worst case computation of a semantic module of $\mathcal{T}$ is polynomial in the size of $\mathsf{sig}(\mathcal{T})$.

5 Checking safeness of a module reuse

Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-lite$_\mathcal{F}$ or DL-lite$_\mathcal{R}$ KB, and let $\mathcal{T}_\Gamma$ be a (possibly robust) semantic module of $\mathcal{T}$ w.r.t. a signature $\Gamma \subseteq \mathsf{sig}(\mathcal{T})$.

The Safe Reuse Checking (SRC) algorithm (Algorithm 2) checks whether a reuse $\mathcal{K}' = \langle \mathcal{T}', \mathcal{A}' \rangle$ of $\mathcal{T}_\Gamma$ w.r.t. $\mathcal{K}$ is safe. Its correctness relies on the use of closure (item *3.* of Proposition 5) for checking entailment. Note that, for any Tbox $\mathcal{T}$ and any signature $\Gamma \subseteq \mathsf{sig}(\mathcal{T})$, $\mathsf{ESM}(\mathcal{T}, \Gamma, \text{false}, \text{false})$ returns a Tbox which is by construction equal to its closure. Because of the lines 7–13 of Algorithm 1, it may not be the case for $\mathsf{ESM}(\mathcal{T}, \Gamma, RQA, RCC)$ when $RQA = \text{true}$. Therefore, comparing the closure of $\mathcal{T}'$ with the *closure* of $\mathsf{ESM}(\mathcal{T} \cup \mathcal{T}', \mathsf{sig}(\mathcal{T}') \backslash \mathsf{sig}^+(\mathcal{T}_\Gamma), RQA, RCC)$ is necessary in line 5 of Algorithm 2 for checking the last item of Definition 6.

Checking safeness of a reuse with Algorithm 2 is polynomial in the size of $\mathsf{sig}(\mathcal{T} \cup \mathcal{T}')$ because of the polynomial time and size of the closure construction (Proposition 5).

Algorithm 2: The SRC algorithm
$\mathsf{SRC}(\mathcal{K}', \Gamma, \mathcal{K}, RQA, RCC)$
Input: a KB $\mathcal{K}' = \langle \mathcal{T}', \mathcal{A}' \rangle$ that is a reuse of the semantic module $\mathcal{T}_\Gamma$ of a DL-lite$_\mathcal{F}$ or DL-lite$_\mathcal{R}$ KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ w.r.t. $\Gamma \subseteq \mathsf{sig}(\mathcal{T})$, and two booleans RQA and RCC denoting respectively whether $\mathcal{T}_\Gamma$ is robust to query answering and/or consistency checking
Output: true if $\mathcal{K}'$ is safe, false otherwise
(1) **if** $\mathsf{sig}(\mathcal{T}) \cap (\mathsf{sig}(\mathcal{T}') \backslash \mathsf{sig}(\mathcal{T}_\Gamma)) \neq \emptyset$
(2) **return** false
(3) **if** $cl(\mathcal{T}) \neq \mathsf{ESM}(\mathcal{T} \cup \mathcal{T}', \mathsf{sig}(\mathcal{T}), \text{false}, \text{false})$
(4) **return** false
(5) **if** $cl(\mathcal{T}') \neq cl(\mathsf{ESM}(\mathcal{T} \cup \mathcal{T}', \mathsf{sig}(\mathcal{T}') \backslash \mathsf{sig}^+(\mathcal{T}_\Gamma), RQA, RCC))$
(6) **return** false
(7) **return** true

6 Conclusion

Compared to the existing works on module extraction, we go a step further by considering the use of extracted modules to develop local KBs. Our work is meaningful in an ontology-based approach for data access and integration. In such a setting, it may be interesting to complete the local answers to queries by answers obtained from a reference global Abox. For doing so in a consistent manner, we have defined the notion of *safe reuse* of a local KB developed from a module extracted from an existing Tbox associated to an existing Abox. We have shown the interest of two properties of module robustness for obtaining global answers in a safe and efficient way, namely

the robustness to query answering and the novel notion of *robustness to consistency checking*.

The semantic modules introduced in this paper generalize both the modules obtained by extracting a subset of a Tbox w.r.t. selected relations or by forgetting concepts or roles.

In contrast with the recent work on extracting modules from DL-lite ontologies [12], we focus on the DL-lite$_{\mathcal{F}}$ and DL-lite$_{\mathcal{R}}$ fragments for which query answering by reformulation guarantees to find *all* the answers. We provide polynomial algorithms for extracting minimal and robust semantic modules. Compared to the algorithm developed by [9] for acyclic $\mathcal{EL}$ ontologies, our approach handles *cyclic* DL-lite$_{\mathcal{F}}$ or DL-lite$_{\mathcal{R}}$ ontologies, while keeping the data consistency and query answering reducible to database queries [3].

In the next future, we plan to evaluate in practice our approach, in particular to compare the size of the modules extracted by our algorithm to the results provided by [4]. We also plan to apply our algorithm for checking the safeness of a reuse on a real case related to the anatomy domain for which the ontology MyCorporisFabrica (www.mycorporisfabrica.org) has been developed manually as a reuse of the reference FMA ontology (sig.biostr.washington.edu/projects/fm).

References

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[2] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[3] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The *dl-lite* family. *JAR*, 39(3):385–429, 2007.

[4] B. Cuenca Grau, I. Horrocks, Y. Kazakov, and U. Sattler. Extracting modules from ontologies: A logic-based approach. In *OWLED*, 2007.

[5] B. Cuenca Grau, I. Horrocks, Y. Kazakov, and U. Sattler. Just the right amount: extracting modules from ontologies. In *WWW*, 2007.

[6] R. Demolombe and P. Pozos Parra. An extension of sol-resolution to theories with equality. In *IJCAR*, 2001.

[7] S. Ghilardi, C. Lutz, and F. Wolter. Did I damage my ontology? a case for conservative extensions in description logics. In *KR*, 2006.

[8] K. Inoue. Linear resolution for consequence finding. *AI*, 2-3(56), 1992.

[9] B. Konev, C. Lutz, D. Walther, and F. Wolter. Semantic modularity and module extraction in description logics. In *ECAI*, 2008.

[10] B. Konev, D. Walther, and F. Wolter. Forgetting and uniform interpolation in extensions of the description logic EL. In *Description Logics*, 2009.

[11] B. Konev, D. Walther, and F. Wolter. Forgetting and uniform interpolation in large-scale description logic terminologies. In *IJCAI*, 2009.

[12] R. Kontchakov, L. Pulina, U. Sattler, T. Schneider, P. Selmer, F. Wolter, and M. Zakharyaschev. Minimal module extraction from DL-Lite ontologies using QBF solvers. In *IJCAI*, 2009.

[13] R. Kontchakov, F. Wolter, and M. Zakharyaschev. Modularity in DL-lite. In *DL*, 2007.

[14] F. Lin and R. Reiter. Forget it! In *AAAI Fall Symposium on Relevance*, 1994.

[15] C. Lutz, D. Walther, and F. Wolter. Conservative extensions in expressive description logics. In *IJCAI*, 2007.

[16] A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Linking data to ontologies. *J. Data Semantics*, 10, 2008.

[17] M. Y. Vardi. The complexity of relational query languages. In *STOC*, 1982.

[18] K. Wang, Z. Wang, R. W. Topor, J. Z. Pan, and G. Antoniou. Concept and role forgetting in ALC ontologies. In *ISWC*, 2009.

[19] Z. Wang, K. Wang, R. W. Topor, and J. Z. Pan. Forgetting concepts in DL-Lite. In *ESWC*, 2008.